#2
A.U.
6/20/01

Docket No. 205127US2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Kazuyoshi KOHNO, et al. GAU:

SERIAL NO: New Application EXAMINER:

FILED: Herewith

FOR: DESIGN VERIFICATION METHOD, DESIGN VERIFICATION DEVICE FOR MICROPROCESSORS, AND PIPELINE SIMULATOR GENERATION DEVICE

REQUEST FOR PRIORITY

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

SIR:

☐ Full benefit of the filing date of U.S. Application Serial Number, filed, is claimed pursuant to the provisions of **35 U.S.C. §120**.

☐ Full benefit of the filing date of U.S. Provisional Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §119(e)**.

■ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of **35 U.S.C. §119**, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY	APPLICATION NUMBER	MONTH/DAY/YEAR
JAPAN	2000-087411	March 27, 2000

Certified copies of the corresponding Convention Application(s)

■ are submitted herewith

☐ will be submitted prior to payment of the Final Fee

☐ were filed in prior application Serial No. filed

☐ were submitted to the International Bureau in PCT Application Number .
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.

☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and

(B) Application Serial No.(s)

- ☐ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.

Marvin J. Spivak
Registration No. 24,913

22850
Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 10/98)

655

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２０００年 ３月２７日

出 願 番 号
Application Number: 特願２０００－０８７４１１

出 願 人
Applicant (s): 株式会社東芝

CERTIFIED COPY OF PRIORITY DOCUMENT

２０００年１２月２２日

特 許 庁 長 官
Commissioner, Patent Office 及 川 耕 造

出証番号 出証特２０００－３１０５７５０

【書類名】　特許願

【整理番号】　4HA99Z073

【提出日】　平成12年 3月27日

【あて先】　特許庁長官殿

【国際特許分類】　G06F 7/00

【発明の名称】　マイクロプロセッサの設計検証方法及びマイクロプロセッサの設計検証装置及びパイプラインシミュレータ生成装置

【請求項の数】　3

【発明者】

【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝マイクロエレクトロニクスセンター内

【氏名】　河野　和義

【発明者】

【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝マイクロエレクトロニクスセンター内

【氏名】　水野　淳

【特許出願人】

【識別番号】　000003078

【氏名又は名称】　株式会社　東芝

【代理人】

【識別番号】　100083806

【弁理士】

【氏名又は名称】　三好　秀和

【電話番号】　03-3504-3075

【選任した代理人】

【識別番号】　100068342

【弁理士】

【氏名又は名称】　三好　保男

【選任した代理人】

【識別番号】　100100712

【弁理士】

【氏名又は名称】　岩▲崎▼　幸邦

【選任した代理人】

【識別番号】　100100929

【弁理士】

【氏名又は名称】　川又　澄雄

【選任した代理人】

【識別番号】　100108707

【弁理士】

【氏名又は名称】　中村　友之

【選任した代理人】

【識別番号】　100095500

【弁理士】

【氏名又は名称】　伊藤　正和

【選任した代理人】

【識別番号】　100101247

【弁理士】

【氏名又は名称】　高橋　俊一

【選任した代理人】

【識別番号】　100098327

【弁理士】

【氏名又は名称】　高松　俊雄

【手数料の表示】

【予納台帳番号】　001982

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　マイクロプロセッサの設計検証方法及びマイクロプロセッサの設計検証装置及びパイプラインシミュレータ生成装置

【特許請求の範囲】

【請求項１】　計算機が解釈可能に記述されたマイクロプロセッサのパイプライン仕様に基づいて前記計算機により検証プログラムを生成し、生成された検証プログラムと前記マイクロプロセッサのＲＴＬ記述に基づいて前記ＲＴＬ記述のシミュレーションを実行し、

また、前記パイプライン仕様に基づいて前記計算機によりパイプラインシミュレータを生成し、前記検証プログラムと生成されたパイプラインシミュレータに基づいてパイプラインシミュレーションを実行し、

前記実行されたＲＴＬ記述のシミュレーション結果と前記パイプラインシミュレーション結果との比較を行い、比較結果に基づいてマイクロプロセッサのパイプライン動作を検証することを特徴とするマイクロプロセッサの設計検証方法。

【請求項２】　計算機が解釈可能に記述されたマイクロプロセッサのパイプライン仕様を入力する入力手段と、

前記入力手段により入力された前記パイプライン仕様に基づいて、前記計算機によりパイプラインシミュレータを生成するシミュレータ生成手段と、

前記入力手段により入力された前記パイプライン仕様に基づいて、前記計算機により検証プログラムを生成するプログラム生成手段と、

前記プログラム生成手段により生成された検証プログラムと前記マイクロプロセッサのＲＴＬ記述に基づいて、ＲＴＬ記述のシミュレーションを実行するＲＴＬシミュレーション実行手段と、

前記プログラム生成手段により生成された検証プログラムと前記シミュレータ生成手段により生成されたパイプラインシミュレータに基づいて、パイプラインシミュレーションを実行するパイプラインシミュレーション実行手段と、

前記ＲＴＬシミュレーション実行手段により実行されたシミュレーションの結果と前記パイプラインシミュレーション実行手段により実行されたシミュレーションの結果を比較し、比較結果に基づいてマイクロプロセッサのパイプライン動

作を検証する比較手段と
を有することを特徴とするマイクロプロセッサの設計検証装置。

【請求項３】　計算機が解釈可能に記述されたマイクロプロセッサのパイプライン仕様を入力する入力手段と、

前記入力手段により入力された前記パイプライン仕様に基づいて、前記計算機によりパイプラインシミュレータを生成するシミュレータ生成手段と
を有することを特徴とするパイプラインシミュレータ生成装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、マイクロプロセッサのＲＴＬ記述の論理検証を行なうマイクロプロセッサの設計検証方法及びマイクロプロセッサの設計検証装置及びパイプラインシミュレータ生成装置に関する。

【０００２】

【従来の技術】

従来、この種の検証を行う方法としては、例えば図３のフローチャートに示す様な手順が知られていた。図３において、破線で示す部分は、プログラムに基づいた計算機による自動生成ではなく、人手によってなされる処理である。日本語や英語等の自然言語で記述されたパイプライン仕様に基づいて検証しようとするプロセッサのＲＴＬ（レジスタ転送レベル）記述を人手により作成する（ステップＳ１０１）。また、パイプライン仕様からシミュレータのソースプログラムを人手により作成し（ステップＳ１０２）、作成されたソースプログラムとパイプライン仕様には依存しないシミュレータのソースプログラム（ステップＳ１０３）とからパイプライン・シミュレータを生成する（ステップＳ１０４）。さらに、パイプライン仕様から検証項目を人手により作成し（ステップＳ１０５）、検証項目から複数の検証プログラムを作成する（ステップＳ１０６）。

【０００３】

次に、プロセッサのＲＴＬ記述と検証プログラムとからＲＴＬシミュレータによりＲＴＬシミュレーションを行い（ステップＳ１０７）、また検証プログラム

を用いてパイプラインシミュレータによりパイプライン・シミュレーションを行い（ステップＳ１０８）、ＲＴＬシミュレーションの結果（ステップＳ１０９）とパイプライン・シミュレーションの結果（ステップＳ１１０）とを比較し（ステップＳ１１１）、パイプライン動作の正否（PASS／FAIL）を判定する（ステップＳ１１２）。

【０００４】

このような検証手順においては、以下に説明するような不具合を招いていた。この不具合の多くはパイプライン仕様が曖昧であることに起因していた。上記従来の検証方法において、パイプライン仕様は日本語、英語等の作業者が容易に理解できる自然言語で記述されていた。このため、本来厳密に定義されるべきパイプライン仕様が、仕様を記述する人間の能力に左右されることになっていた。このような状況は、自然言語で記述されたパイプライン仕様を解釈する場合にも当てはまる。このことにより、ＲＴＬ実装者、パイプライン・シミュレータ実装者、検証プログラム作成者にパイプライン仕様の誤解が生ずる可能性があった。

【０００５】

ＲＴＬ検証を行なう場合に、パイプライン・シミュレータ実装者のパイプライン仕様の誤解は検証効率に大きくな影響を及ぼす。なぜならば、パイプライン・シミュレータはリファレンス・モデルとして使用されるものであり、ＲＴＬでのシミュレーション結果と比較する際に絶対に正しい結果を出すものであることを前提に検証作業が行なわれるからである。

【０００６】

検証プログラム作成者のパイプライン仕様の誤解は、論理検証洩れの問題をはらんでいる。論理検証洩れのままマイクロプロセッサを製造し、実機を用いたアプリケーション・ソフトでの論理検証で誤りが発見された場合には、最悪の場合に、論理設計からやり直す必要が生ずる。また、マイクロアーキテクチャの変更に即座に対応できないことも問題である。１個の命令セットアーキテクチャに対して、様々なマイクロアーキテクチャが存在し、さらにマイクロアーキテクチャの仕様は設計段階で変更されることがある。このため、従来の方法では、マイクロアーキテクチャの仕様が設計段階で変更される度に、パイプライン・シミュレ

ータおよび検証プログラムを人手で変更する必要があった。

【０００７】

【発明が解決しようとする課題】

以上説明したように、マイクロプロセッサのＲＴＬ記述における従来の論理検証方法にあっては、自然言語で記述された同一のパイプライン仕様に基づいて、ＲＴＬ実装、パイプラインシミュレータ実装、検証プログラム作成が行なわれていたため、パイプライン仕様の解釈の食い違いによる誤りが発生していた。さらに、検証プログラムはパイプライン仕様から人手により生成されていたため、検証洩れが起こりやすかった。また、ＲＴＬシミュレーションの結果と検証プログラムに基づくパイプラインライン・シミュレーションの結果が不一致の場合に、いずれに不具合があるのかの究明が困難であった。さらに、プロセッサの性能向上の目的等でマイクロアーキテクチャ仕様の変更が行なわれた場合には、容易に対応することが困難であった。

【０００８】

そこで、この発明は、上記に鑑みてなされたものであり、その目的とするところは、パイプライン動作を正確かつ充分に検証できるマイクロプロセッサの設計検証方法及びマイクロプロセッサの設計検証装置及びパイプラインシミュレータ生成装置を提供することにある。

【０００９】

【課題を解決するための手段】

上記目的を達成するために、課題を解決する第１の手段は、計算機が解釈可能に記述されたマイクロプロセッサのパイプライン仕様に基づいて前記計算機により検証プログラムを生成し、生成された検証プログラムと前記マイクロプロセッサのＲＴＬ記述に基づいて前記ＲＴＬ記述のシミュレーションを実行し、また、前記パイプライン仕様に基づいて前記計算機によりパイプラインシミュレータを生成し、前記検証プログラムと生成されたパイプラインシミュレータに基づいてパイプラインシミュレーションを実行し、前記実行されたＲＴＬ記述のシミュレーション結果と前記パイプラインシミュレーション結果との比較を行い、比較結果に基づいてマイクロプロセッサのパイプライン動作を検証することを特徴とす

る。

【００１０】

第２の手段は、計算機が解釈可能に記述されたマイクロプロセッサのパイプライン仕様を入力する入力手段と、前記入力手段により入力された前記パイプライン仕様に基づいて、前記計算機によりパイプラインシミュレータを生成するシミュレータ生成手段と、前記入力手段により入力された前記パイプライン仕様に基づいて、前記計算機により検証プログラムを生成するプログラム生成手段と、

前記プログラム生成手段により生成された検証プログラムと前記マイクロプロセッサのＲＴＬ記述に基づいて、ＲＴＬ記述のシミュレーションを実行するＲＴＬシミュレーション実行手段と、前記プログラム生成手段により生成された検証プログラムと前記シミュレータ生成手段により生成されたパイプラインシミュレータに基づいて、パイプラインシミュレーションを実行するパイプラインシミュレーション実行手段と、前記ＲＴＬシミュレーション実行手段により実行されたシミュレーションの結果と前記パイプラインシミュレーション実行手段により実行されたシミュレーションの結果を比較し、比較結果に基づいてマイクロプロセッサのパイプライン動作を検証する比較手段とを有することを特徴とする。

【００１１】

第３の手段は、計算機が解釈可能に記述されたマイクロプロセッサのパイプライン仕様を入力する入力手段と、前記入力手段により入力された前記パイプライン仕様に基づいて、前記計算機によりパイプラインシミュレータを生成するシミュレータ生成手段とを有することを特徴とする。

【００１２】

【発明の実施の形態】

以下、図面を用いてこの発明の実施形態を説明する。

【００１３】

図１はこの発明の一実施形態に係るマイクロプロセッサの設計検証方法の手順を示すフローチャートである。本発明は、マイクロプロセッサのパイプライン仕様記述を入力し、その仕様を用いてサイクル・アキュレートなシミュレータ（以下、パイプライン・シミュレータと記す）を生成する機能を有し、また、入力さ

れたパイプライン仕様を用いて、パイプライン動作モデルを生成する機能を有し、さらに、生成されたパイプライン動作モデルを用いて、パイプライン動作を検証するための必要かつ充分な検証プログラムを生成する機能を有する。このような機能を有することにより、生成された検証プログラムを用いてマイクロプロセッサのＲＴＬ記述のシミュレーションを行なうと同時に同一の検証プログラムを用いてパイプライン・シミュレータを実行し、両者の結果を機械的に比較することにより洩れなく効率よくマイクロプロセッサのＲＴＬ記述の論理検証を行なうようにしている。

【００１４】

まず、図１を参照して、この発明の一実施形態に係るマイクロプロセッサの設計検証方法について説明する。図１の破線の部分は、図３と同様に人手によってなされる処理を示す。図１において、自然言語ではなく、計算機で解釈可能に記述されたパイプライン仕様（詳細は後述する）に基づいて検証しようとするプロセッサのＲＴＬ記述を人手により作成する（ステップＳ１）。また、上記パイプライン仕様に基づいて計算機によりシミュレータのソースプログラムを生成し（ステップＳ２）、生成されたシミュレータのソースプログラムと、パイプライン仕様に依存しないシミュレータのソースプログラム（ステップＳ３）とからパイプライン・シミュレータを生成する（ステップＳ４）。さらに、パイプライン仕様から計算機によりパイプラインライン動作モデルを生成し（ステップＳ５）、生成されたパイプライン動作モデル（ステップＳ６）に基づいて計算機により検証項目を生成し（ステップＳ７）、生成された検証項目（ステップＳ８）とステップＳ６のパイプライン動作モデルとに基づいて検証プログラムを計算機により生成し（ステップＳ９）、検証プログラム（ステップＳ１０）を得る。

【００１５】

パイプライン仕様記述からパイプライン動作モデルを生成し、更にパイプライン動作モデルを用いて検証プログラムを生成する方法に関しては、例えば、特願平１１－７４１１８号に開示されている技術を使用することにより実現可能である。パイプライン動作モデルの生成ステップＳ５では、パイプラインを構成する各ステージの接続関係を有向グラフで表現したパイプライン構造グラフの作成、

構造ハザードによるストール条件の決定、データ・ハザードによるストール条件の決定、各パイプライン・ステージの命令の組み合わせを状態とみなした有限状態機械（finite state machine）の生成が行われる。検証項目は、各パイプライン・ステージの状態の組み合わせで表現され、検証項目生成ステップＳ７では、構造ハザードが起こる各パイプライン・ステージの状態、データ・ハザードが起こる各パイプライン・ステージの状態を列挙して検証項目を生成する。検証プログラム生成ステップＳ９では、パイプライン動作モデルおよび検証項目から検証項目の各要素を検証するための命令列（検証プログラム）を自動生成する。これらを用いた命令列の自動生成は、ＢＢＤを用いて容易に実現可能である。ＢＤＤ（Ｂｉｎａｒｙ　Ｄｅｃｉｓｉｏｎ　Ｄｉａｇｒａｍ）とは、論理関数のグラフ表現の一種であり、論理関数をコンパクトに表現することができる。前述の有限状態機械をＢＢＤを用いて表現し、初期状態から到達可能な状態集合を計算することにより命令列を生成する。

【００１６】

次に、プロセッサのＲＴＬ記述と検証プログラム（ステップＳ１０）とからＲＴＬシミュレータによりＲＴＬシミュレーションを行い（ステップＳ１１）、また検証プログラムに基づいてパイプラインラインシミュレータによりパイプライン・シミュレーションを行い（ステップＳ１２）、ＲＴＬシミュレーションの結果（ステップＳ１３）とパイプライン・シミュレーションの結果（ステップＳ１４）とを比較し（ステップＳ１５）、パイプライン動作の正否（PASS／FAIL）を判定する（ステップＳ１６）。

【００１７】

このようなパイプライン動作の検証手順において、パイプラインの仕様を作成する際に、パイプライン仕様を記述する記述言語を定義し、その定義に従ってパイプライン仕様を記述する。

【００１８】

次に、パイプライン仕様の具体的な記述方法について説明する。本パイプライン仕様には暗黙の制御ルールがあり、特に指定しなくても命令がストールする場合があるものとする。暗黙の制御ルールは次の２つである。

【００１９】

（１）同一名ステージに同時に複数の命令は存在できない。

【００２０】

（２）Ｉｎ－Ｏｒｄｅｒ（インオーダー）で命令が完了する。

【００２１】

同一名のステージは、物理的にも同一のハードウェアを意味する。あるステージで命令がストールする場合には、それに続く命令もストールするという制御は、この制御ルールにより実現される。ステージの機能が同じでも、対象となるリソースが異なるため、同時に実行可能なステージがある場合には、パイプライン仕様の記述では異なるステージ名をつけなければならない。

【００２２】

以下では、代表的な命令に対してこの実施形態のパイプライン仕様の記述を例示する。以下に説明するパイプライン仕様の記述において、パイプラインの各ステージは、時間的な流れの順に、Ｆステージ（Ｆｅｔｃｈ　ｓｔａｇｅ：命令フェッチステージ）、Ｄステージ（Ｄｅｃｏｄｅ　ｓｔａｇｅ：命令デコードステージ）、Ｅステージ（Ｅｘｅｃｕｔｉｏｎ　ｓｔａｇｅ：命令実行ステージ）、Ｍステージ（Ｍｅｍｏｒｙ　ｓｔａｇｅ：メモリへの書き込み／読み出しステージ）、Ｗステージ（Ｗｒｉｔｅ　ｂａｃｋ　ｓｔａｇｅ：レジスタへの書き込みステージ）とする。まず、ＡＬＵ命令のパイプライン仕様記述の一例を示す。バイパスの動作は次のように記述する。例えば、ＡＬＵパイプラインでは、Ｗステージで計算結果をレジスタへ書き込むが、バイパスによりＥステージおよびＭステージの直後で値が参照可能であるとした場合は、次のように記述する。

【００２３】

ＡＬＵ．Ａｃｃｅｓｓ（Ｅ）；

ＡＬＵ．Ｆｏｒｗａｒｄｉｎｇ（Ｅ＋Ｍ）；

ＲＥＧ．ＷｒｉｔｅＢａｃｋ（Ｗ）；

一般的には次のような記述となる。

【００２４】

演算ユニット．Ａｃｃｅｓｓ（ステージ（の並び））

演算ユニット．Ｆｏｗｒａｒｄｉｎｇ（ステージ（の並び））

レジスタ．Ｒｅａｄ（ステージ）

レジスタ．ＷｒｉｔｅＢａｃｋ（ステージ）

Ａｃｃｅｓｓはリソース競合判定のために使われる。この指定によりリソース競合によるストールが実行される。Ｒｅａｄはオペランドが揃うタイミングの判断基準となる。シミュレータ内部ではオペランドがすべてそろったステージで命令を実際に実行する。Ｒｅａｄが指定されていない命令では、発行ステージ（現在はＤ）の直後のステージで命令を実行している。ＦｏｒｗａｒｄｉｎｇやＷｒｉｔｅＢａｃｋの指定はデータ利用可能の判定で使用される。判定アルゴリズムを以下に示す。

【００２５】

（データ利用可能判定アルゴリズムの一例）

ｉｆ（Ｆｏｒｗａｒｄｉｎｇが定義されいる）ｔｈｅｎ

Ｆｏｒｗａｒｄｉｎｇ

並びの先頭ステージが現サイクルで実行終了、またはすでに実行終了なら利用可能。

【００２６】

ｅｌｓｅ　ｉｆ（ＷｒｉｔｅＢａｃｋが定義されている）ｔｈｅｎ

ＷｒｉｔｅＢａｃｋステージが現サイクルで実行終了、またはすでに実行終了なら利用可能

ｅｌｓｅ利用不可能

ｅｎｄｉｆ

次に、ストール動作は以下のように記述される。例えば汎用レジスタＲＥＧに対してＲＡＷハザードを検出した場合に、Ｄステージでストールするパイプライン動作を定義する場合は、次のように記述する。

【００２７】

ＳｔａｌｌＳｔａｇｅ（ＲＡＷ，ＲＥＧ）＝Ｄ；

一般的には次のように記述する。ハザードの要因としては、ＲＡＷ，ＷＡＷ，ＣＯＮＦＬＩＣＴ（リソース競合）が指定できる。

【００２８】

ＳｔａｌｌＳｔａｇｅ（ハザード要因、対象とするリソース）

＝ストールするステージ；

ストール定義の記述があった場合は、シミュレータは以下に示すアルゴリズムでパイプライン動作を行う。

【００２９】

（シミュレータのストール制御アルゴリズムの一例）

Ｓ：ストールするステージ、Ｒ：対象となるリソース、ＴＩ：Ｒを利用する先行命令とする。命令が現サイクルでＳステージにあるとき、ハザード要因がＲＡＷのとき、ＴＩがＲに書き込むべき値がその時点で利用可能でないならストールする。ハザード要因がＷＡＷのとき、ＴＩがＲへの書き込みを終えてなければストールする。ハザード要因がＣＯＮＦＬＩＣＴのとき、ＴＩがＲへアクセス中ならストールする。上記以外はストールしない。

【００３０】

ここで、先行命令がレジスタに書き込む値が利用可能かどうかは、先行命令のバイパスとレジスタ書き込みステージの定義により決まる。それについては次に説明する。このようにして定義されたＡＬＵ命令のパイプライン仕様記述を以下に示す。

【００３１】

（ＡＬＵ命令のパイプライン仕様記述例）

／／ＡＬＵパイプライン

／／ＲＡＷハザードが検出された時は、Ｄステージでストールする。

【００３２】

／／ＥステージでＡＬＵ演算を行う。レイテンシは１である。

【００３３】

／／ＥまたはＭステージでＡＬＵ演算結果をフォーワードする。

【００３４】

／／Ｗステージでレジスタに書き込む。

【００３５】

```
ＡＬＵＰｉｐｅ：：ＡＬＵＰｉｐｅ（）
 {
  ｎａｍｅ＝“ＡＬＵ＿Ｐｉｐｅｌｉｎｅ”；
  Ｓｔａｇｅｓ＝Ｆ＋Ｄ＋Ｅ＋Ｍ＋Ｗ；
  ／／リソースとデータフローの関係
  ＲＥＧ．Ｒｅａｄ（Ｄ）；
  ＡＬＵ．Ａｃｃｅｓｓ（Ｅ）；
  ＡＬＵ．Ｆｏｒｗａｒｄｉｎｇ（Ｅ＋Ｍ）；
  ＲＥＧ．ＷｒｉｔｅＢａｃｋ（Ｗ）；
  ／／ハザードによりストールさせるステージ
  ＳｔａｌｌＳｔａｇｅ（ＲＡＷ，ＲＥＧ）＝Ｄ；
}
```

次に、ロード命令に関するパイプライン仕様記述を例示する。ロード命令はＭステージで命令をロードしバイパスする。そのため、直後のデータ依存のある命令は１サイクルストールする。この動作を実現するためのパイプライン仕様記述を以下に示す。

【００３６】

（ロードパイプラインの仕様記述例）

```
／／ロードパイプライン
／／ＲＡＷハザードが検出された時は、Ｄステージでストールする。
```

【００３７】

```
／／ＥステージでＡＬＵ演算（アドレス計算）を行う。
```

【００３８】

```
／／Ｍステージでデータのロードに成功するとＭステージでフォーワードする。
```

【００３９】

```
／／Ｗステージでレジスタに書き込む。
```

【００４０】

```
ＬＤＰｉｐｅ：：ＬＤＰｉｐｅ（）
 {
```

```
    name＝“Load_Pipeline”；
    Stages＝F＋D＋E＋M＋W；
    //リソースとデータフローの関係
    REG. Read（D）；
    ALU. Access（E）；
    MMU. Load（M）；
    REG. WriteBack（W）；
    //ハザードによりストールさせるステージ
    StallStage（RAW, REG）＝D；
}
```

次に、以下に示すような命令列を考える。

【００４１】

```
    lw  $1.（$10）；
   and  $2. $1；
```

ａｎｄ命令は上述したＡＬＵのパイプラインの定義に従うものとし、ｌｗは上述したロードパイプラインの定義に従うものとする。この命令列は以下のようなパイプライン動作をする。小文字はストールを表す。

【００４２】

```
 lw  $1.（$10）F  D  E  M  W        ロードパイプライン
and  $2. $1       F  d  D  E  M  W  ＡＬＵパイプライン
                     ↑  ↑
                    (a) (b)
```

時刻（ａ）でＤステージに来たので、ＲＡＷハザードをチェックする。先行のｌｗとの間にＲＡＷ依存関係があるので、ｌｗの結果が利用可能かどうかを調べる。ｌｗは時刻（ａ）でＥステージにいるが、ロードパイプラインはＦｏｒｗａｒｄｉｎｇがＭステージから始まるのでまだ結果利用可能ではない。そのため、ａｎｄ命令は（ａ）ではストールする。次の時刻（ｂ）でもａｎｄ命令はＤステージなのでＲＡＷハザードをチェックする。同じくｌｗの間にＲＡＷ依存関係があるのでｌｗの結果が利用可能かどうかを調べる。今回はｌｗがＭステージを実行

することになっているので１ｗの結果が利用可能であり、ハザードは検出されずａｎｄ命令はストールしない。

【００４３】

上述したロードパイプラインの定義において、以下に示す行を無効にすると、ロードパイプラインの結果利用可能ステージがＷステージになり、同じ例ではａｎｄ命令がＤステージが２サイクルストールし、バイパス機能がない状況を設定できる。

【００４４】

ＭＭＵ．Ｆｏｒｗａｒｄｉｎｇ（Ｍ）；

次に、乗算命令のパイプライン仕様を例示する。乗算パイプラインの定義を以下に示す。

【００４５】

（乗算パイプラインの仕様記述例）

```
／／乗算命令が従う
／／オペランドのＲＥＧに対するＲＡＷハザードが検出された時は、
／／Ｄステージでストールする。
```

【００４６】

```
／／ＭＡの演算はＥ、Ｍステージで行う。
```

【００４７】

```
／／レイテンシは２
／／ＨＩ／ＬＯへの書き込みは、Ｍステージで行う。
```

【００４８】

```
／／ＭＵＬＤＩＶに対するリソース競合があってもストールしない
／／先行命令がＤＩＶならＤＩＶをキャンセルしてそのまま実行
／／先行命令がＭＵＬなら先行のＭＵＬをつぶしてそのまま実行

 ＭＵＬＴＰｉｐｅ：：ＭＵＬＴＰｉｐｅ（）
{
 ｎａｍｅ＝“Ｍｕｌｔｉｐｌｙ＿Ｐｉｐｅｌｉｎｅ”；
```

```
//out-of-order完了を実現するためステージ名を変更
Stages=F+D+C(2)+X;
//リソースとデータフローの関係
REG. Read(D);
MULDIV,. Access(C);
HI, WRiteBack(X);
LO, WriteBack(X);
//ハザードによりストールさせるステージ
StallStage(RAW, REG)=D;
//先行mul/divはキャンセルさせる
F. MakeFlush(Mult);
F. MakeFlush(Div);
CompletionOrder=OutofOrder//Out-of-
Order完了。
```

【００４９】

}

乗算命令が乗除算器（ＭＵＬＤＩＶ）で実行されている間は、他の命令がＡＬＵを実行してもかまわない。しかし乗算パイプラインのＭＵＬＤＩＶ実行ステージをＡＬＵ実行スライドと同じＥステージにすると、シミュレータの制御ルールにより、ＭＵＬＤＩＶをアクセスしている間は他の命令はＡＬＵをアクセスできず、ストールしてしまう。これを避けるためにＣいう新しいＭＵＬＤＩＶ用のステージを定義している。

【００５０】

乗算命令を実行中、依存関係のない後続の命令は先に終了する。すなわちＯｕｔ－ｏｆ－Ｏｒｄｅｒ完了であり、これを指定するのが上記乗算パイプラインの定義における、

```
CompletionOrder=OutofOrder;/Out-of-
Order完了。
```

【００５１】

の行である。これを無効にすると後続命令はたとえ乗算命令と依存関係がなくても乗算命令が終了するまでストールする。また、同一ステージに複数の命令が移れないという制御ルールから、乗算命令がＣステージを終了しない間では、後続のＣステージを使う命令（乗算と除算）は手前のＤステージでストールする。これは多サイクルステージでレイテンシを調整した場合に、デフォルトではスループットは多サイクルステージのサイクル数以上になることを意味する。スループットをレイテンシ以下にしたい場合は、ステージ並びを増やす方法をとる。リソース競合などを定義しなければスループットは１となるが、スループットを１以外にするには、パイプライン定義中で、ｔｈｒｏｕｇｈｐｕｔ＝３のように指定する。

【００５２】

先行の乗算や除算命令が終了しないうちに次の乗算や除算命令が投入された場合には、先行命令をキャンセルする制御を定義したのが、

Ｆ．ＭａｋｅＦｌｕｓｈ（Ｍｕｌｔ）；

Ｆ．ＭａｋｅＦｌｕｓｈ（Ｄｉｖ）；

である。これは自身がＦステージの時点でパイプライン中に他の乗算や除算命令がある場合に、それをキャンセルすることを意味する。一般的には次の書式である。

【００５３】

ステージＭａｋｅＦｌｕｓｈ（命令カテゴリー）

最後に、分岐命令のパイプライン仕様を例示する。常に分岐不成立として先読みし、分岐命令がＥステージで条件を判定し、分岐を行うときはその時点でのＦステージとＤステージにある命令をキャンセルするというパイプライン仕様記述を以下に示す。

【００５４】

（分岐パイプラインの仕様記述例）

／／分岐パイプライン

／／分岐／ジャンプ、リピート、リターン命令が従う

／／ＲＡＷハザードが検出された時は、Ｄでストールする。

【００５５】

//ＥステージでＡＬＵ演算（条件判定）と、ＰＣの上書きを行う。

【００５６】

```
ＢＲＰｉｐｅ：：ＢＲＰｉｐｅ（）
{
  ｎａｍｅ＝“Ｂｒａｎｃｈ＿Ｐｉｐｅｌｉｎｅ”；
  Ｓｔａｇｅｓ＝Ｆ＋Ｄ＋Ｅ＋Ｍ＋Ｗ；
  ＰｒｅｄｉｃｔｅｄＶａｌｕｅ＝ＮｏｔＴａｋｅｎ；//常に分岐不成立と予測
  //リソースとデータフローの関係
  ＲＥＧ．Ｒｅａｄ（Ｄ）；
  ＡＬＵ．Ａｃｃｅｓｓ（Ｅ）；
  ＬＰ．Ｒｅａｄ（Ｄ）；
  ＥＰＣ．Ｒｅａｄ（Ｄ）
  //ハザードによりストールさせるステージ
  ＳｔａｌｌＳｔａｇｅ（ＲＡＷ，ＲＥＧ）＝Ｄ；
  //分岐予測がはずれたとき（分岐命令がＥステージでｔａｋｅｎとなったとき）
  //Ｆ、Ｄステージの命令を次のサイクルへ移る時にフラッシュする。
      【００５７】
  Ｅ．ｓｅｔＣｏｎｔｒｏｌ（＆ｂｒＨｚｄ）；
}
```

以下の記述文で常に分岐不成立の予測を指定している。

【００５８】

```
ＰｒｅｄｉｃｔｅｄＶａｌｕｅ＝ＮｏｔＴａｋｅｎ；//常に分岐不成立と予測
```

ＰｒｅｄｉｃｔｅｄＶａｌｕｅにはＴａｋｅｎとＮｏｔＴａｋｅｎだけが指定できる。現状では複雑な分岐予測には対応できない。

【００５９】

```
E. setControl (&brHzd) ;
```

はＥステージで分岐予測とそれに伴う動作を行うことを意味している。ｂｒＨｚｄは分岐予測の結果を返すクラスオブジェクトである。

【００６０】

（分岐予測時のパイプラインの仕様記述例）

```
void BRPipe::ActionBranchpredict
{
//分岐予測が外れた。
```

【００６１】

```
if (brHzd. BranchPredictResult (PredictedValue. PC) ==false)
//F、Dステージの命令をフラッシュ
F. Flush () ;
D. Flush () ;
}
}
```

Ｅステージで実行される動作の実体は、上述したように記述される。分岐予測が外れた場合には、Ｆ、Ｄステージの命令を取り出してキャンセルしている。このように、分岐予測の方向、分岐予測を行うステージ、およびそのときのキャンセルすべき命令の指定は容易に行うことができる。

【００６２】

本実施形態では、パイプライン仕様記述としてＣ++の関数で記述例を示したが、必要なデータの入力を要求する専用のユーザインタフェイスを用意して、得られたデータをＣ++関数等に変換することは容易に実現可能である。

【００６３】

このように、上記実施形態においては、計算機が解釈可能に記述された同一のパイプライン仕様を元に、ＲＴＬ実装、パイプラインシミュレータの生成、検証プログラムの生成が行なわれるため、パイプライン仕様の食い違いによる誤りを防ぐことができる。また、検証プログラムの生成はパイプライン仕様から計算機

により自動生成されるため、仕様が正しく記述されていれば検証洩れを防止することができる。さらに、パイプライン仕様を元に人手で実装する部分はＲＴＬ記述であるため、ＲＴＬシミュレーションの結果とパイプラインシミュレーションの結果とを比較してシミュレーション結果が合わない場合には、ＲＴＬ実装に問題があると判断することができ、バグの原因を究明する効率が向上する。また、性能向上の目的等でマイクロアーキテクチャ仕様の変更が行なわれた場合でも、容易に対応することができる。

【００６４】

図２はこの発明の一実施形態に係るマイクロプロセッサの設計検証装置の構成を示す図である。図２において、この実施形態のマイクロプロセッサの設計検証装置は、図１に示す設計検証手順にしたがってマイクロプロセッサのパイプラインライン動作の検証を行うものであり、前述したように計算機が解釈可能な記述言語によって記述されたパイプライン仕様を入力するパイプライン仕様入力手段１と、シミュレータのソースプログラムを入力するシミュレータのソースプログラム入力手段２と、パイプライン仕様入力手段１により入力されたパイプライン仕様を保持するパイプライン仕様保持手段３と、入力されたパイプラインライン仕様ならびにシミュレータのソースプログラムに基づいてパイプラインシミュレータを生成するパイプラインシミュレータ生成手段４と、パイプラインシミュレータ生成手段４により生成されたパイプラインシミュレータを保持するパイプラインシミュレータ保持手段５と、パイプラインシミュレータ生成手段４により生成されたパイプラインシミュレータ又はパイプラインシミュレータ保持手段５で保持されたパイプラインシミュレータに基づいてパイプラインシミュレータを実行するパイプラインシミュレータ実行手段６と、パイプライン仕様保持手段３に保持されたパイプライン仕様に基づいてパイプラインライン動作モデル、検証項目を生成した後検証プログラムを生成する検証プログラム生成手段７と、検証プログラム生成手段７によって生成された検証プログラムを保持する検証プログラム保持手段８と、ＲＴＬデータを入力するＲＴＬデータ入力手段９と、ＲＴＬデータ入力手段９により入力されたＲＴＬデータを保持するＲＴＬデータ保持手段１０と、ＲＴＬ保持手段１０に保持されたＲＴＬデータをに基づいてＲＴＬシミ

ュレーションを実行するＲＴＬシミュレーション実行手段１１と、ＲＴＬシミュレーション実行手段１１により実行されたＲＴＬシミュレーションの結果とパイプラインシミュレータ実行手段６により実行されたシミュレーション結果を比較して、パイプライン動作の正否（PASS／FAIL）を判定するシミュレーション結果比較手段１２を備えて構成されている。なお、パイプラインシミュレータ保持手段５は、パイプラインシミュレータ生成手段４により生成されたパイプラインシミュレータを再利用する際には保持したパイプラインシミュレータをパイプラインシミュレータ実行手段６に与えるので、パイプラインシミュレータ生成手段４によりパイプラインシミュレータを毎回生成する場合には、パイプラインシミュレータ保持手段５は不要となる。

【００６５】

このような構成において、計算機で解釈可能な記述言語で記述されたパイプライン仕様に基づいて生成された検証プログラムを用いてマイクロプロセッサのＲＴＬ記述のシミュレーションを行なうと同時に、同一の検証プログラムを用いてパイプライン・シミュレータを実行し、両者の結果を機械的に比較することにより洩れなくかつ効率よくマイクロプロセッサのＲＴＬ記述の論理検証を行なうようにしている。

【００６６】

このような実施形態においては、上記実施形態と同様の効果を得ることが可能である。

【００６７】

【発明の効果】

以上説明したように、この発明によれば、計算機が解釈可能に記述されたパイプライン仕様を共通に入力して、計算機により検証項目を自動生成し、ＲＴＬシミュレーションならびにパイプラインシミュレーションを行うようにしたので、充分かつ正確にパイプラインライン動作を効率よく検証することが可能となる。

【図面の簡単な説明】

【図１】

この発明の一実施形態に係るマイクロプロセッサの設計検証方法の手順を示す

フローチャートである。

【図２】

この発明の一実施形態に係るマイクロプロセッサの設計検証装置の構成を示す図である。

【図３】

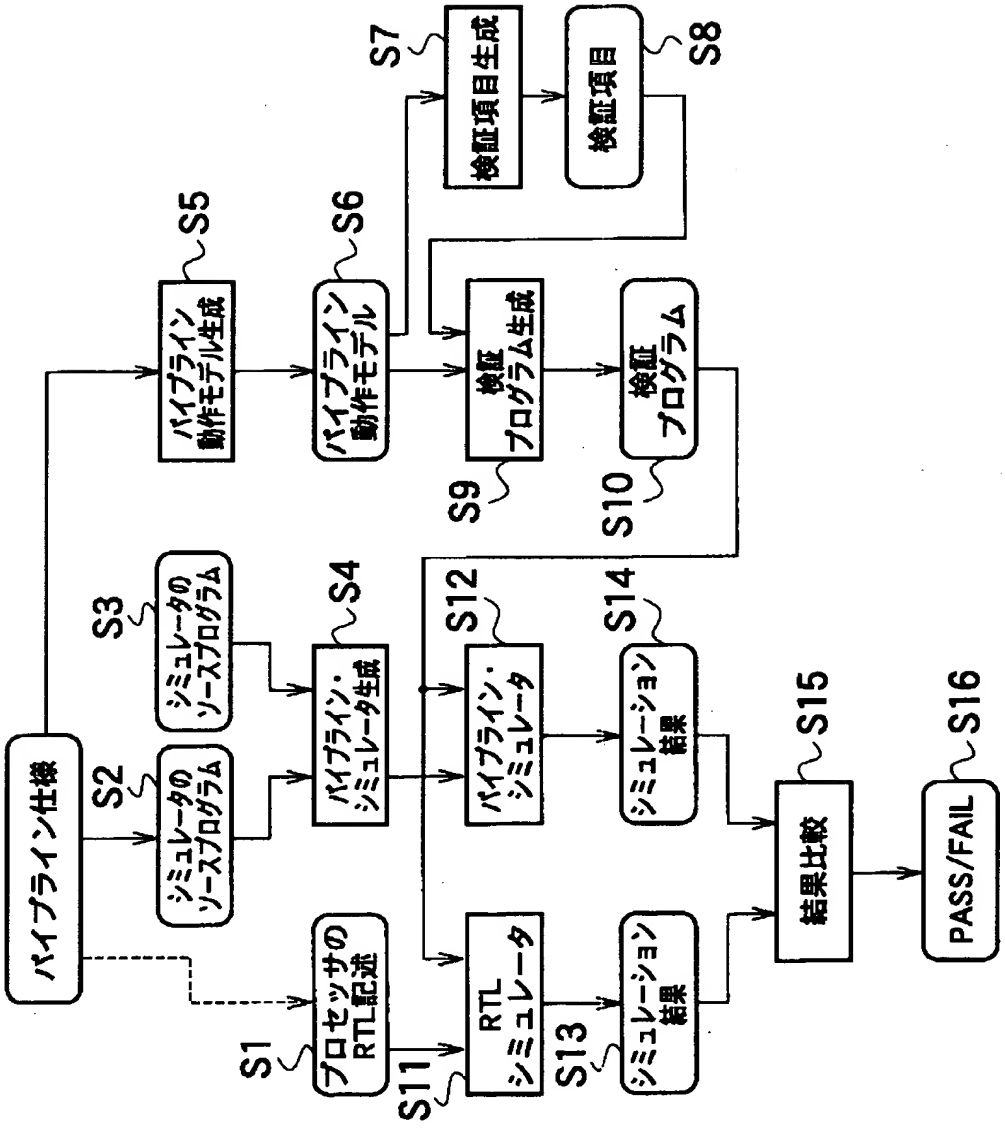
従来のマイクロプロセッサの設計検証方法の手順を示すフローチャートである。

【符号の説明】

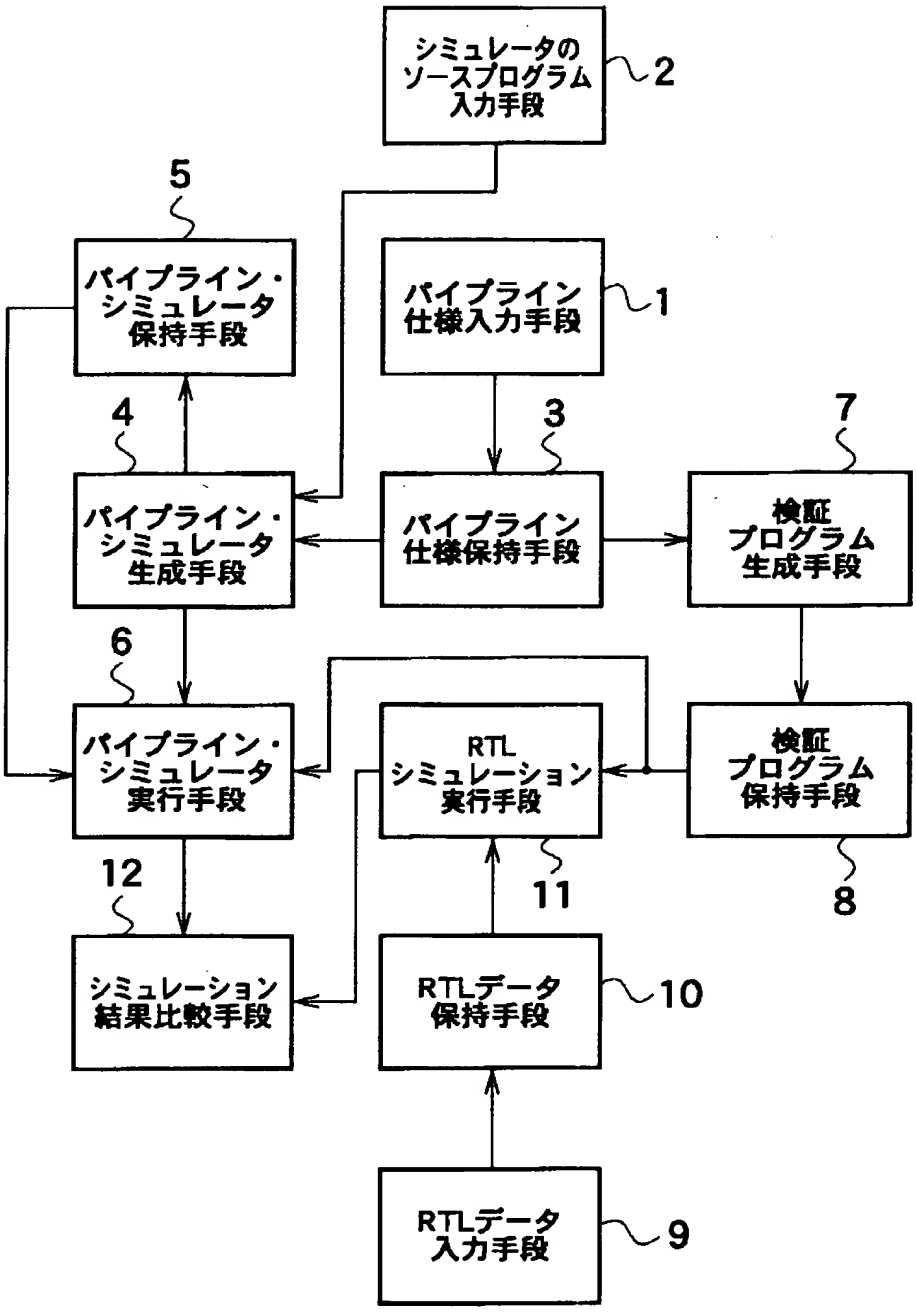
１　パイプライン仕様入力手段
２　シミュレータのソースプログラム入力手段
３　パイプライン仕様保持手段
４　パイプラインシミュレータ生成手段
５　パイプラインシミュレータ保持手段
６　パイプラインシミュレータ実行手段
７　検証プログラム生成手段
８　検証プログラム保持手段
９　ＲＴＬデータ入力手段
１０　ＲＴＬデータ保持手段
１１　ＲＴＬシミュレーション実行手段
１２　シミュレーション結果比較手段

【書類名】　　図面

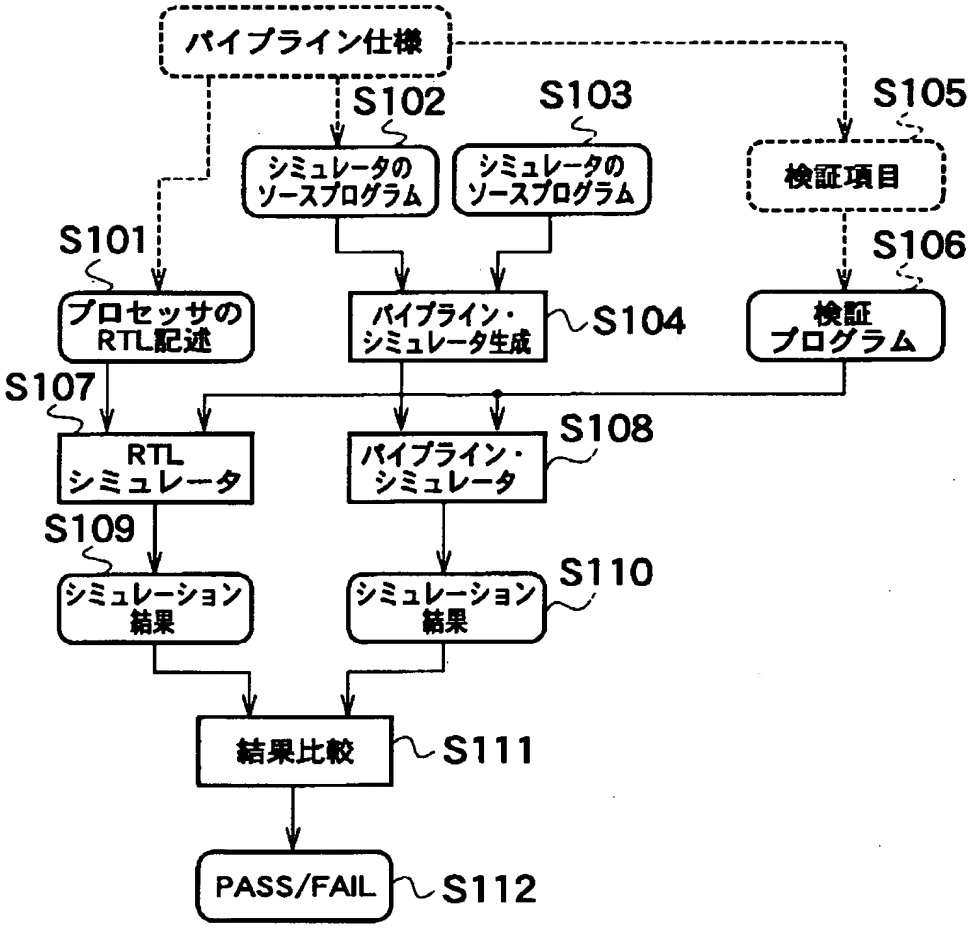
【図１】


パイプライン仕様
S1
プロセッサのRTL記述
S2
シミュレータのソースプログラム
S3
シミュレータのソースプログラム
S4
パイプライン・シミュレータ生成
S5
パイプライン動作モデル生成
S6
パイプライン動作モデル
S7
検証項目生成
S8
検証項目
S9
検証プログラム生成
S10
検証プログラム
S11
RTLシミュレータ
S12
パイプライン・シミュレータ
S13
シミュレーション結果
S14
シミュレーション結果
S15
結果比較
S16
PASS/FAIL

【図２】


シミュレータのソースプログラム入力手段
2
5
パイプライン・シミュレータ保持手段
パイプライン仕様入力手段
1
4
3
7
パイプライン・シミュレータ生成手段
パイプライン仕様保持手段
検証プログラム生成手段
6
パイプライン・シミュレータ実行手段
RTLシミュレーション実行手段
検証プログラム保持手段
12
11
8
シミュレーション結果比較手段
RTLデータ保持手段
10
RTLデータ入力手段
9

【図３】


パイプライン仕様
S102
S103
S105
シミュレータのソースプログラム
シミュレータのソースプログラム
検証項目
S101
S106
プロセッサのRTL記述
パイプライン・シミュレータ生成
S104
検証プログラム
S107
S108
RTLシミュレータ
パイプライン・シミュレータ
S109
S110
シミュレーション結果
シミュレーション結果
結果比較
S111
PASS/FAIL
S112

【書類名】　　要約書

【要約】

【課題】　この発明は、マイクロプロセッサのパイプライン動作を正確かつ充分に効率よく検証することを課題とする。

【解決手段】　この発明は、計算機が解釈可能に記述されたパイプライン仕様に基づいて、パイプラインシミュレータ、検証プログラムを生成し、検証プログラムとＲＴＬ記述に基づいて実行されたＲＴＬ記述のシミュレーションの結果と、検証プログラムとパイプラインシミュレータに基づいて実行されたパイプラインシミュレーションの結果に基づいてパイプライン動作を検証するように構成される。

【選択図】　　図１

出　願　人　履　歴　情　報

識別番号　　〔０００００３０７８〕

１．変更年月日　　１９９０年　８月２２日
［変更理由］　新規登録
住　所　　神奈川県川崎市幸区堀川町７２番地
氏　名　　株式会社東芝